OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Digital Engineering Project Report

Color Based Camera Tracking of Multiple Quadcopters

Supervisors:
Dr. Sanaz Mostaghim
Dipl.-Inform. Christoph Steup

Submitted by:
Jawad Ahmad
Kanwal Jahan
Yusra Shakeel

Table of Contents

Written by: Yusra Shakeel

Chapter 1:

1 Introduction

For the evaluation of the performance of algorithms written for quadcopters, it is important that we track their movement in an isolated environment, and use that information as the reference. The aim of this project is to perform 2D position and orientation estimation of copters using colored LEDs and a camera system that doesn't rely on the on-board copter software. Instead, a completely decoupled tracking system is designed to accurately estimate copter pose using image processing. The investigated scenario is completely indoor within an arena inside a lab. The camera is centrally mounted on the ceiling over the arena which provides sharp colored images.

1.1 Key Words

Pose; position along with orientation,

Robot Operating System(ROS); With the help of this OS certain fields of data can be subscribed and published by a node,

Blob; LEDs mounted on quadcopters are detected as blobs during image processing,

Key points; No. of LEDs detected,

Cluster; LEDs belonging to one quadcopter are grouped together,

Region of Interest; Area in a frame where possibility of finding a quadcopter is maximum.

1.2 Problem Statement

The goal of this project is to publish copters' complete pose information i.e.; with the positions (x / y-coordinates), the orientation (yaw-angle) and the associated ID.

1.3 System Overview

1.3.1 Hardware

A quadcopter is a multirotor which is lifted and propelled with the help of four rotors/fans. Two of them rotate clockwise and two anti-clockwise. Independent variation of the speed of each rotor is used to achieve control. By changing the speed of each rotor it is possible to specifically generate a desired total thrust. [1]

In our configuration we have two rotors of color green(front) and two of orange(back).

Every quadcopter has four LEDs installed on the top, two colors are used to distinguish between front and back side of a copter. In our configuration the side with two green LEDs is considered front and with red LEDs is labelled back. In addition to that, to distinguish between multiple copters there is another LED in the center. Each copter has a different color for this 5th LED.

1.3.2 Software

The environment used to capture video stream, process the data and publish the pose information is ROS, Distribution 9: Jade Turtle, running on linux. Python, version 2.7.11 is used to write the application code, while the dependencies involved are Numpy, version 1.11.0 for vector manipulation and OpenCV 3.0, to perform image processing operations. The software has following sections:

1. Blob detection
2. LED color detection
3. Cluster formation
4. Pose estimation (Center and Angle)
5. Region of interest calculation

Chapter 2:

2 Single copter as the starting point

Written by: Yusra Shakeel

The goal of the first phase of development was accurate pose estimation of a single copter flying in the arena. Test data consisting of a 30-seconds long 60-FPS video steam (a ROS. bag file) was collected, in which a single copter was captured rotating around one point, without moving. This was done in order to eliminate complexities brought by a laterally moving copter. Routines for blob detection, LED color detection and pose estimation were tested on static images that were randomly extracted from the bag file. Later they were tested on the whole stream.

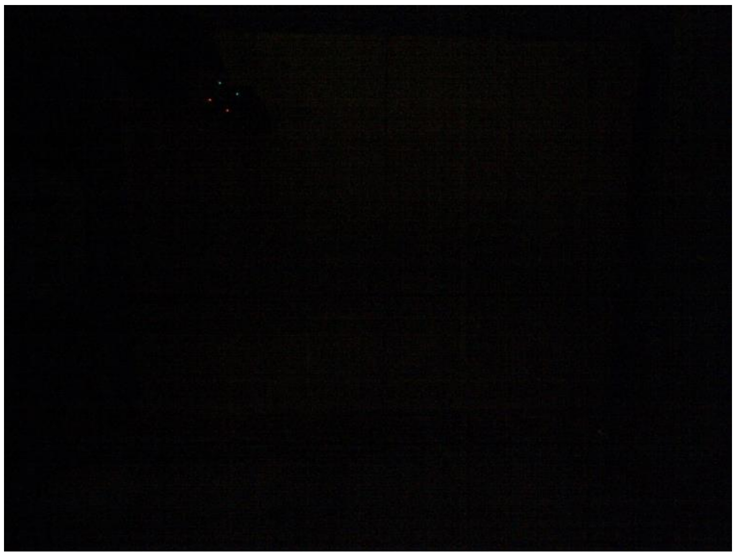

Figure 2.1: Raw image with one copter in it.

2.1 Blob detection:

Written by: Yusra Shakeel

Implemented by: Yusra Shakeel

Because the LEDs were the brightest portions of the captured images, we decided to use "Blob Detection" to detect them. Since each image consists of a copter that has four LEDs on it, detecting LEDs consequently allowed us to detect the copter as a single object.

This routine performs basic pre-processing on the selected images. This pre-processing involved gray-scaling and then thresholding to create binary images. This removed all of the noise, except the bright LEDs. The copter appears as a cluster of four spots. The blob detector provided us what we call the "key-points" of the copter.

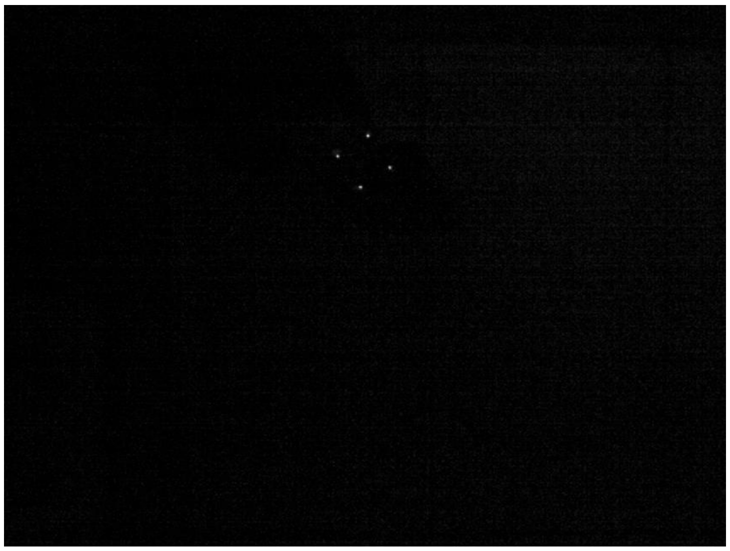

Figure 2.2: Result of blob detection.

2.2 LED color detection:

Written by: Kanwal Jahan

Implemented by: Kanwal Jahan

Once we have the LED locations, a 3x3 pixel area around each LED in the original image is analyzed to guess its color. Keeping the approach simple, mean values of Red, Green and Blue channels are calculated in that region. The three values are then compared against one another. The channel with the highest mean value is taken as the LED color.

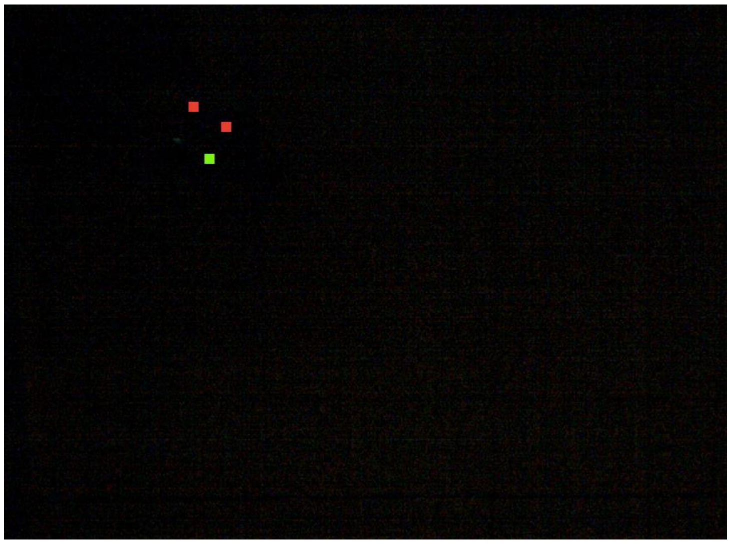

Figure 2.3: Result of LED color detection.

2.3 Pose estimation

The next logical step after successfully getting position and LED color detection is the pose estimation of the copter.

2.4 Center detection

Written by: Yusra Shakeel

Implemented by: Yusra Shakeel

For a trivial case where we have position information of all four LEDs, we can safely say that the geometric center of these LEDs is actually the center of quadcopter itself. But this is not always the case; sometimes not all of the four LEDs are detected. In several images, the fan of the copter and the LEDs align in such a way that the brightness of one or more LEDs falls low enough to be eliminated in thresholding. It is necessary that we have at least three LEDs to be captured to have an accurate pose estimation. So images with less than three LEDs captured are ignored. In software, the point between two different and farthest LEDs in the image is taken as the center of the copter. Turns out, it works equally good for three and four detected LEDs.

2.5 Angle detection

Written by: Yusra Shakeel

Implemented by: Yusra Shakeel

Improved by: Jawad Ahmad

The angle of the copter is calculated as the angle of the perpendicular to the line joining two LEDs of the same color. The perpendicular's head/tail depends on if the two LEDs are front ones (green) or the rear ones (red). At this point, we have the information about position and orientation of the quadcopter in a static image.

.

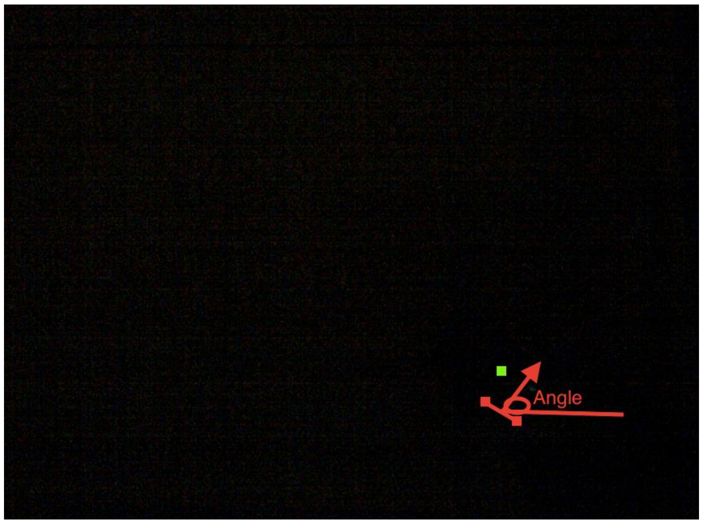

Figure 2.4: Pose detection.

2.6 Tests on video stream

This technique was then tested on the continuous stream from the bag file. It worked as good as it did for the static images.

2.7 Architecture Overview:

Written by: Jawad Ahmad

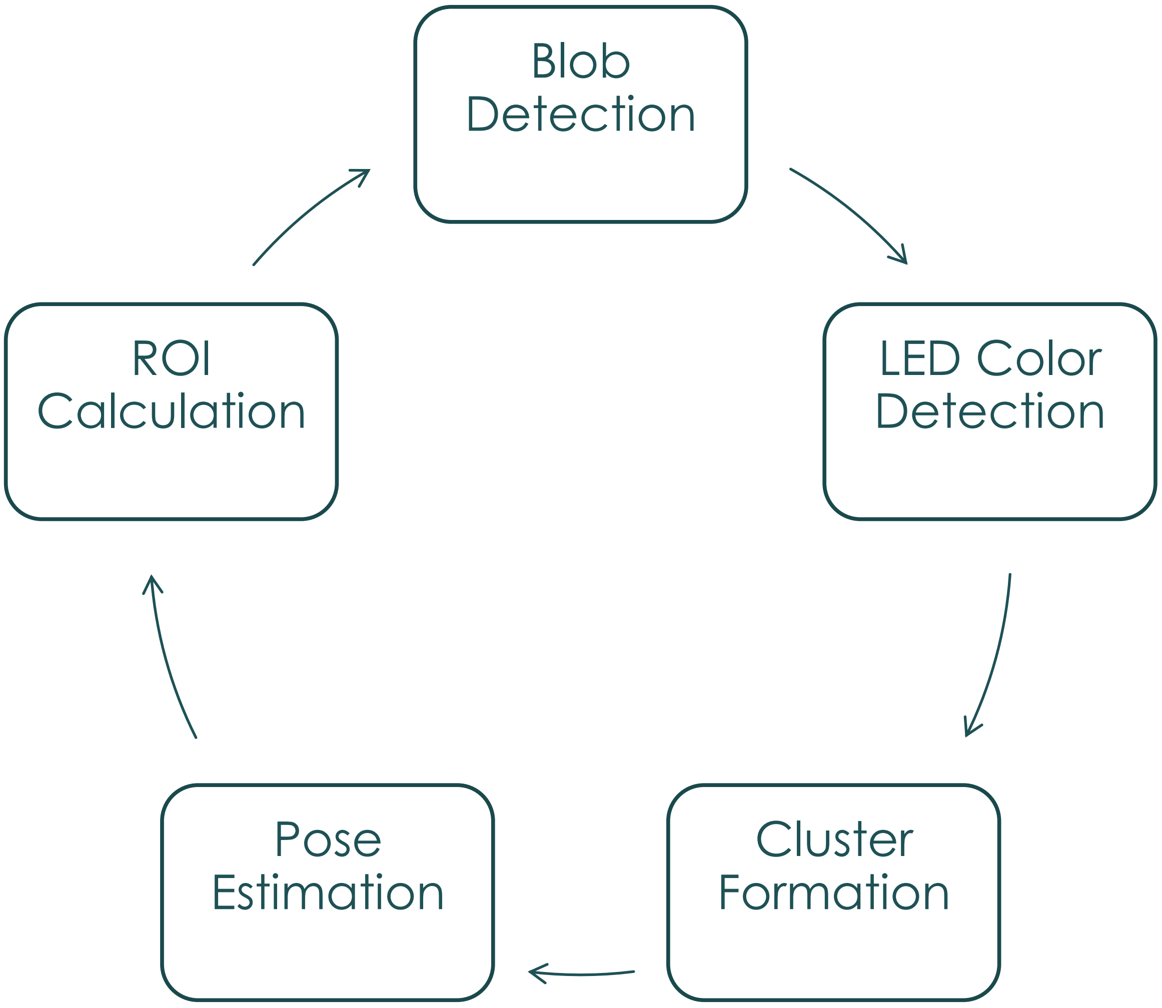

Figure 2.5: Architecture for pose estimation of multiple quadcopter.

Chapter 3:

3 Scaling to multiple copters

Written by: Jawad Ahmad

Implemented by: Jawad Ahmad

This phase involved scaling our pose estimation software for multiple quadcopter scenario. The techniques for blob detection, LED color detection, center and angle estimation remain same except for one problem; How do we know which LED belongs to which copter?

3.1 Cluster Formation

Written by: Jawad Ahmad

Implemented by: Jawad Ahmad

Once we have detected all LEDs on the image for all the visible copters, we have to form clusters of LEDs based on the vicinity to each other. Default search radius for this purpose was safely assumed to be 100 pixels, which worked excellent for the assumed height of copter from the ground. We made the assumption that the copters will never come too close to one another for their LEDs to be closer than 100 pixels. Once we have the clusters formed, we can crop the image around each cluster and apply the estimation methods to each of them separately. Keeping a frame of reference for each cropped part is essential here to translate the returned results to the global frame of reference. This clustering is done once every 60 frames for reasons that are mentioned later.

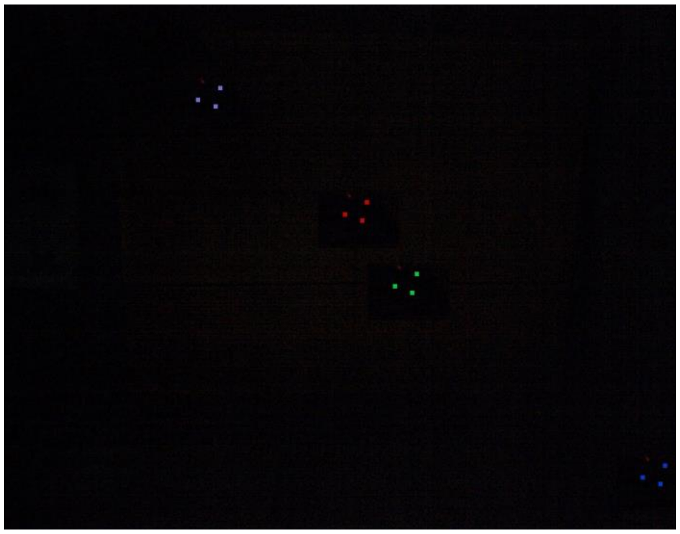

Figure 3.1: Four clustered formed for four quadcopters.

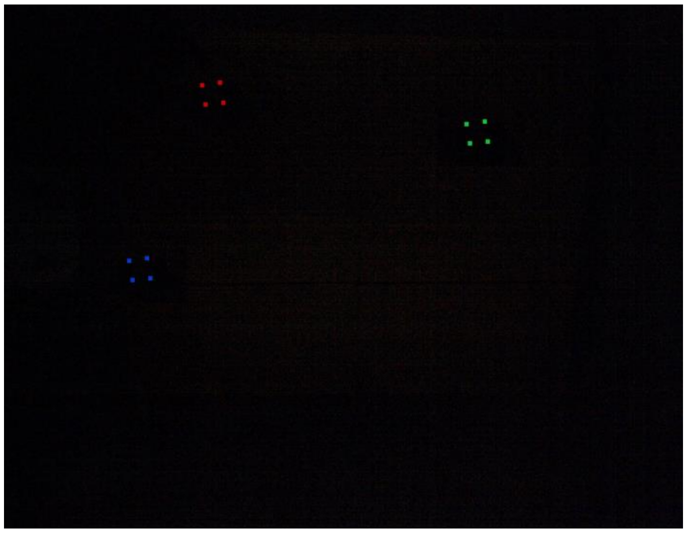

Figure 3.2 : Cluster of one color belongs to one copter.

3.2 Region of Interest

Written by: Jawad Ahmad

Implemented by: Jawad Ahmad

After getting pose information for each copter, an efficient step can be taken to reduce the search area for the next image. That is, to create Regions of Interest (ROI) for each copter. Instead of looking for LEDs all over again in the whole image, we only look for them around the last known location of each copter. This search window is set to be 100x100 pixels in size, centered at the last known center of the copter itself. Depending on the speed of the copters, LEDs may or may not be found in this area. In case they are not, the window is increased again by the same factor, i.e. 200x200 pixels, for the next cycle until the LEDs are found or the window expands to the size of the image itself. If the copter just disappears from the scene, the next cluster formation cycle will eliminate it from the search.

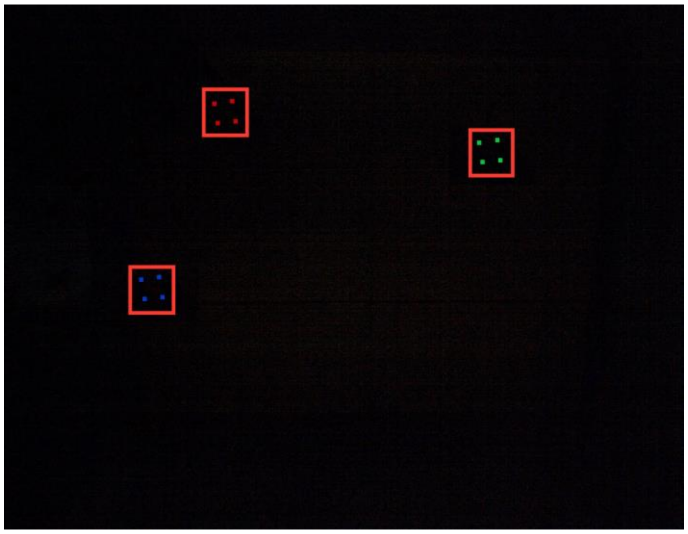

Figure 3.3: Red window is a depiction of 100x100 pixel around each quadcopter.

3.3 Identification of Copters

Written by: Kanwal Jahan
Implemented by: Kanwal Jahan

With the current set up 5 copters can be distinguished from each other. The fifth LED which is installed in the center as a marker can have blue, white, violet, yellow or cyan color.
Every color in the code corresponds to one id. So quadcopters in our configuration can have id 1,4,5,6, and 7. For combination colors like yellow, magenta, cyan and white we see the mean of respective basic color and with carefully tuning a threshold, presence and absence is found.

Table 3.1: Color and respective IDs.

Color	ID
Blue	1
Green	2
Red	3
White	4
Yellow	5
Cyan	6
Magenta	7
Unknown	15

Chapter 4:

4 Results and Discussion

4.1 Results

Data published from each quadcopter contained X, Y coordinate and orientation.

Quadcopter #1: Detected id of marker LED is 4

Sample of the data published for quadcopter detected with ID 4 is as under.

Table 4.1: Data published for quadcopter with ID 4(White LED)

Timestamp (Epoch)	ID	X Coordinate	Y Coordinate	Orientation	Quality (No. of LEDs)
1476723668628490924	4	861	498	348.477011359	0.8
1476723668682657957	4	861	498	348.398487364	0.8
1476723668731122016	4	861	498	348.205796634	1
1476723668788893938	4	861	498	348.477011359	1
1476723668860048055	4	861	498	348.5737984	0.8
1476723668923908948	4	861	498	347.92295391	1
1476723668991442918	4	861	498	348.558591234	1

Data collected over 10 seconds is visually interpreted with the help of graphs below.

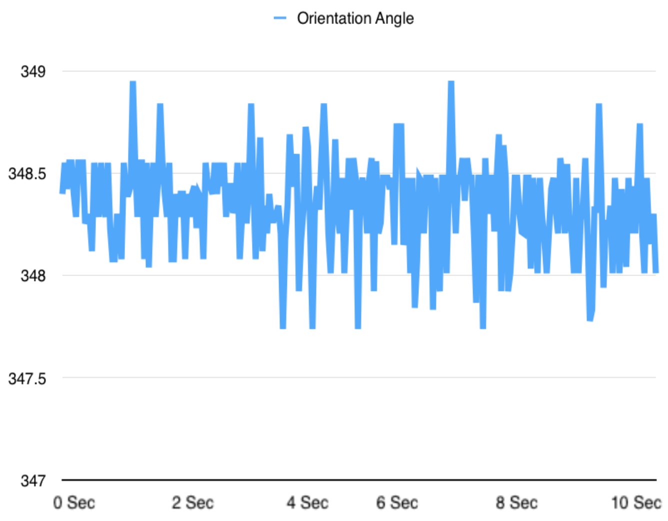

Figure 4.1: Orientation angle in degrees measured for 10 secs.

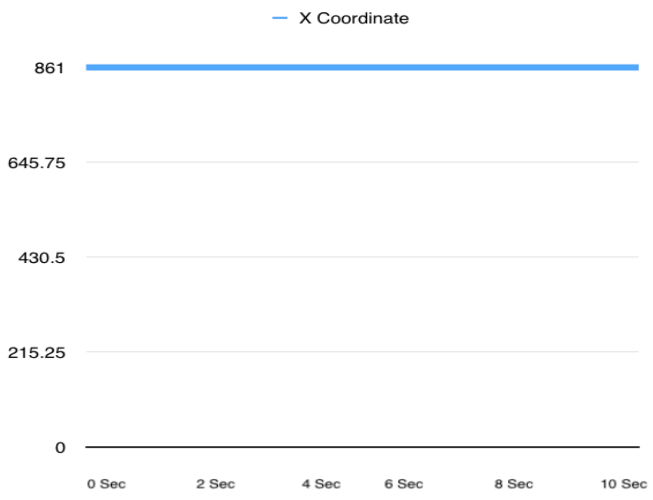

Figure 4.2: X coordinate measured for 10 secs.

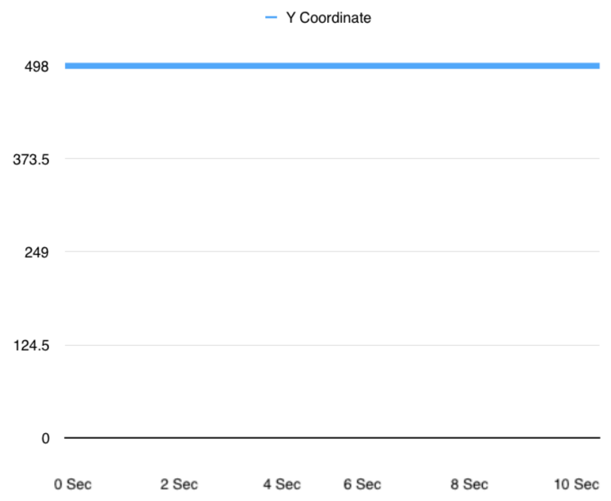

Figure 4.3: Y coordinate measured for 10 secs..

Copter Detection

$$\textbf{Copter detection ratio} = \frac{\textbf{\#of times data published}}{\textbf{Total time stamps}} = \frac{\textbf{818}}{\textbf{2410}} = \textbf{34\%}$$

Mean

x-coordinate = 861

y-coordinate = 498

Orientation = 348.36 degree

Variance

x-coordinate =0

y-coordinate = 0

Orientation = 0.040728 degree

These values show that data points are not spread very far away from the mean value or in other words there is no significant change in position and orientation. Since id 4 belongs to white quadcopter while white copter was actually stationary during the data collection/bag file creation. This shows not only the correct identification of the quadcopter but also smoothness of pose detection too.

Quadcopter #2: Detected ID of marker LED is 1

Sample of the data published for quadcopter detected with ID 1 is listed below.

Table 4.2: Data published for quadcopter with ID 1 (Blues LED).

Timestamp (Epoch)	ID	X Coordinate	Y Coordinate	Orientation	Quality (No. of LEDs)
1476723083449127912	1	761	674	200.772254682	1
1476723083486960887	1	761	674	201.806592234	0.8
1476723083544264078	1	761	674	200.772254682	0.8
1476723083577384948	1	761	674	200.772254682	1
1476723083621581077	1	761	674	202.205998101	0.8
1476723083651519060	1	761	674	200.772254682	1
1476723083698909997	1	761	674	200.772254682	1

Below interpreted data belongs to quadcopter having marker id 1, Data sampled over 1 and 10 seconds is plotted respectively.

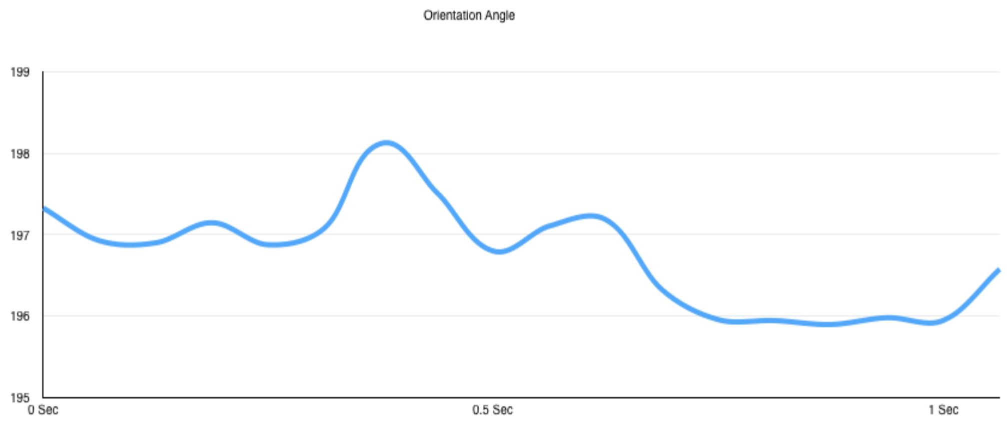

Figure 4.4: Orientation angle measured for 1 sec.

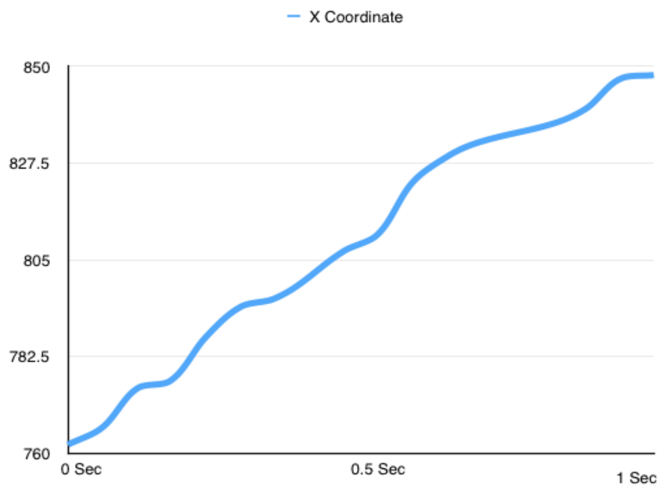

Figure 4.5: X coordinate measured for 1 sec.

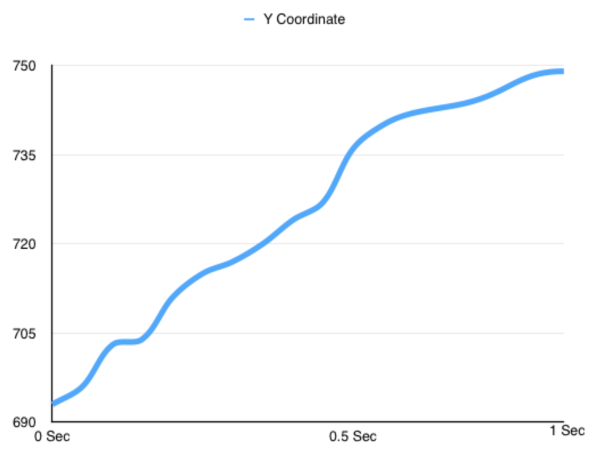

Figure 4.6: Y coordinate measured for 1 sec.

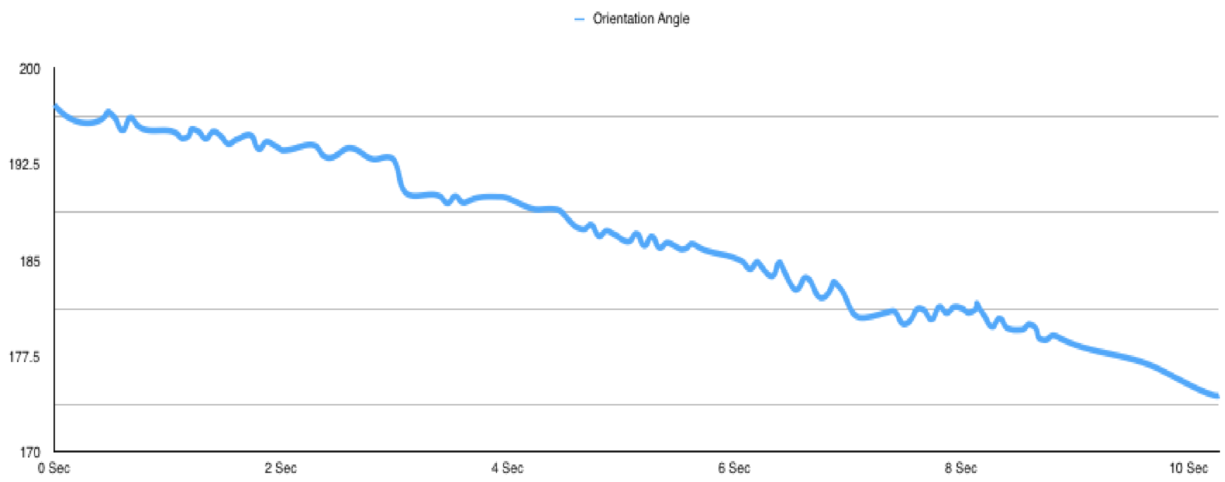

Figure 4.7: Orientation angle measured for 10 secs.

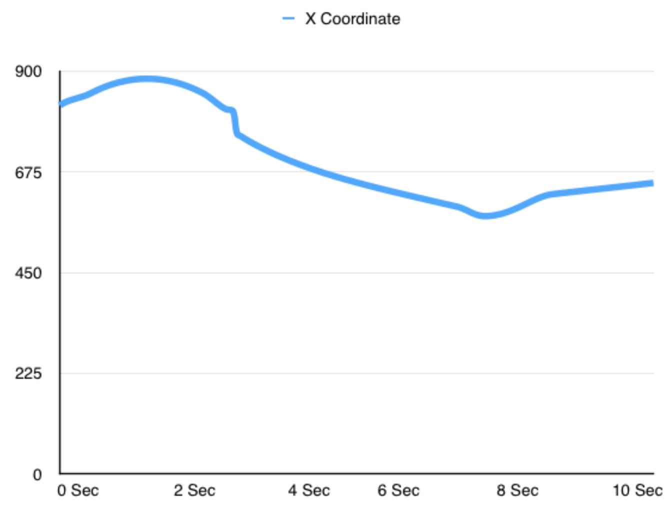

Figure 4.8: X coordinate measured for 10 secs.

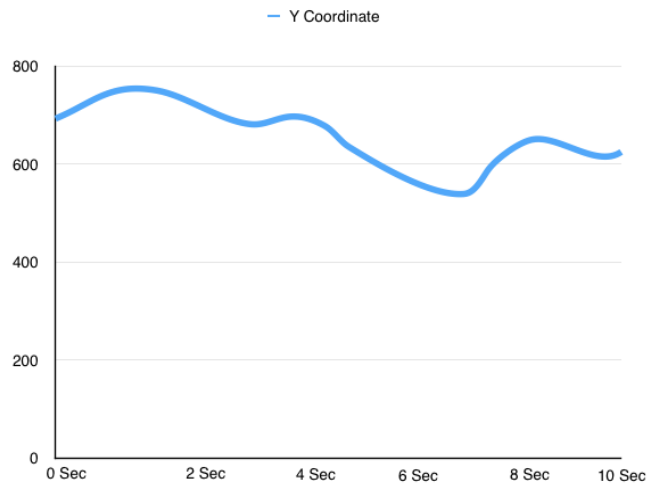

Figure 4.9: Y coordinate measured for 10 secs.

Copter Detection

$$\boldsymbol{Copter\ detection\ ratio = \frac{\#of\ times\ data\ published}{Total\ time\ stamps} = \frac{1046}{2410} = \ \ 43\%}$$

Mean

For 1 sec:

x-coordinate = 807.058

y-coordinate = 724.11

Orientation = 196.76 degree

For 10 secs:

x-coordinate = 710.58

y-coordinate = 659.508

Orientation = 187.038 degree

Variance

For 1 sec:

x-coordinate = 0.410

y-coordinate =0.427

Orientation = 0.430 degree

For 10 secs:

x-coordinate = 10985.50

y-coordinate = 4063.83

Orientation = 48.38 degree

Variance value shows that data of this copter was captured during its flight. ID of the quadcopter id detected 1 which belongs to color blue. During the creation of bag file blue quadcopter was actually flying.

4.2 Evaluation

Written by: Jawad Ahmad

The ID for the copters are detected correctly i.e.; White and Blue. Stationary copter shows ideal low variance values for X, Y coordinates as 0 and orientation is 0.040 degrees. Significantly high variance value of orientation is correctly noted for the moving copter. Code is robust, works well with three detected LEDs too. Successful cluster formation for copters with at least three LEDs detected on them.

The assumption that copters never come closer than 100 pixels is never violated, provided copters are flying in the same height. The ROI of 100x100 pixel is evaluated to perform best. All frames with less than 3 LEDs detected on quadcopter are discarded. Code runs on Mac (Core i5, dual core, 4 GB RAM running in a VM), published pose data ratio is 34% for stationary quadcopter and 43% for moving quadcopter.

Summary

Written by: Jawad Ahmad

We were able to successfully publish the pose information for a number of quadcopters and distinguish between them. The algorithm performs robustly as it works even when all four LEDs are not detected. The software is designed to consume minimum computer resources, and will absolutely be sufficient for 60-FPS processing on a system with medium specifications.

4.3 Future Work

Written by: Jawad Ahmad

Improvements that can be made to the current system involve a better use of custom ROS messages for publishing. Also, prevent collision of search windows while expanding them and transform the software to work as a node in the background.

4.4 Reference

1. Wikipedia on 17-10-2016 https://en.wikipedia.org/wiki/Quadcopter

4.5 Code

https://github.com/ovgu-FINken/CameraTracking/tree/pose